UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/676,373	09/30/2003	Stefan Jesse	09700.0216-00	3224

60668 7590 12/15/2009

SAP / FINNEGAN, HENDERSON LLP
901 NEW YORK AVENUE, NW
WASHINGTON, DC 20001-4413

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
12/15/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/676,373	JESSE ET AL.	
	Examiner	Art Unit	
	TUAN A. VU	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *8/31/09*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3,4,6-10,12,14-18 and 20-26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3,4,6-10,12,14-18 and 20-26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 8/31/09.

As indicated in Applicant's response, claims 1, 10, 18, 23-24 have been amended, claims 2, 5, 11, 13, 19 canceled, and claims 25-26 added. Claims 1, 3-4, 6-10, 12, 14-18, 20-26 are pending in the office action.

Claim Objections

2. Claim 18 is objected to because of the following informalities: the spelling in 'that is a represnation' (line 8) is a clear typo error for which correction is required.

3. The language recited 'tangibly embodied in a information carrier' in claims 1, 10 and 18 does not preclude a carrier wave signal as one example of such carrier, and waveform medium cannot constitute a permissible *statutory category of subject matter* in terms of USC § 101 statute. Correction is strongly required.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 3-4, 6-10, 12, 14-18, 23-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kadel et al, USPubN: 2002/0184401 (hereinafter Kadel), in view of Severin, USPubN: 2005/0005251 (hereinafter Severin) further in view of Yeluripati et al, USPN: 7,086,065 (hereinafter Yeluripati).

As per claim 1, Kadel discloses a computer program product, tangibly embodied in an information carrier, for developing an application, the computer program product being operable to cause data processing apparatus to:

receive a first model in a first language, the first model defining development objects representing building blocks for developing the application, relationships among the development objects, and constraints for developing the application;

generate a set of intermediate objects (e.g. Fig. 5; para 0194-0198, pg. 15; *JavaBeans ... package com.xis.leif.event ... includes interfaces* - para 0210-0213, pg. 16; *domain policy methods ... returned ... procedural components of the metadata and methods* - para 0334, pg. 29), and

generate an API using the set of intermediate objects as inputs (e.g. Add resource class, loads other pluggable services into pluggable manager – Fig. 29; para 0344, pg. 29; *Java interfaces ... instantiate the appropriate bean* -para 0349-0350 pg. 30; *exposing attributes, Java introspection* - para 0129-0135 pg. 9).

Kadel does not explicitly disclose instructions to convert the first model to a second model in a second language, wherein the second language comprises XMI, and generating intermediate objects by parsing the second model using an XML parser.

Kadel discloses deriving Java objects association with XML constructs or schema (para 0297-0306,pg. 23-25), implementing a "Domain Policy" using XIS framework and DSI interface to expose java objects based on metadata relationship information derived for a domain (see Fig. 13; para 0193, 0203, pg. 4; Fig. 14, 36A-B), relationship to implement the required data flow between source and consumer; and further discloses database for assisting the DSI being

represented as extensible markup *schema* (e.g. para 0288-0304, pg. 24; para 0084-0085, pg. 5) to represent said *Domain Policy* attributes or typemetadata, attribute relationship or dependency, or declaration constraints, such that the metadata representing a application domain and Java related beans exposed by the DSI (see para 0311-0312, pg. 26) in XIS framework are represented in this XML schema form, such that the XML schema can in reverse, be **imported back** into a framework (see Kadel: para 0083, pg. 5; *XML schema ... sends to the receiver ... receiver reconstructs the information* - para 0305, pg. 25) its content exposed by the XML-DSI (para 0318, pg. 26) in relation to the database (Fig. 30). Hence the deriving of Java classes based on domain XML and the corresponding UML is suggested (see Kadel: Fig. 13). The interchangeability of XML and UML in terms of mapping of UML constructs into XML schema and vice versa was well-known and disclosed in Severin. Following to this concept of Kadel's UML/XML inter-changeability, Severin discloses UML constructs (Severin: Fig. 4-8), with use of XML metasyntax and XMI methodology (Severin: *XMI* - para 0184, pg. 15) to represent this extensible meta language in terms of definitions, inter-relationships or constraints (e.g. Severin: zero-to-many, metahints, relationship, constraint, datatype – see para 0139-0148, pg. 11-12) reading a persisted model (para 0508, pg. 41) and re-mapping metamodel data and corresponding UML package constructs (MVC para 0107-0108, pg. 8; Fig. 32) to derive underlying Java classes or package (para 0196, pg. 16; para 0107-0108, pg. 8). That is, the well-known W3C XMI methodology in terms of data interchange description -- or a second model -- based on a first model (e.g. UML as in Kadel) including model description language to correlate with XML components is evidenced in Severin 's (XML, XMI - para 0184, pg. 15) where rediscovery based on such XMI model description enable remapping into XML elements which

are derived for further development; e.g. mapping for developing Java objects or classes, APIs for some domain application. Based on the UML constructs and derived class objects as taught in Kadel's use of the DSI approach and XML processor (XML stream 3002, XML Processor – Fig. 30) the tight association between meta-information and the deriving of Java objects from XML model as shown in Kadel and the XMI implementation as taught in Severin to enable rediscover content of a XML model, it would have been obvious for one skill in the art at the time the invention was made to implement Kadel's XML schema as first metamodel so that a transformation to this model yield a XML-compliant model supported via a XMI description language thus exposing the UML instance (see Severin) and underlying Java objects as taught above, because this second model would be used to better collect and identify objects (deriving of UML and underlying Java objects therefrom – see Kadel: Fig. 13) exposed from the first model received in a portable schema stream format using the W3C methodology and its useful techniques supporting this XML/model interchangeability as this is also perceived in Kadel, and Severin.

Nor does Kadel explicitly disclose wherein the API comprises *XML marshalling code and XML schema* to enforce constraints. The relationship between metadata and standard Java API constructs is disclosed in Kadel as reflection of data between domain-describing schemas and said Java or Corba API standards (e.g. reflected Java objects – para 0311, pg. 25, para 0312, pg. 26) from which XML metadata can be implemented to enforce a database of domain requirements (e.g. Table XIV, XVI, XVIII, pg. 26-27), e.g. XML schema so to enforce domain constraints. Hence, Kadel's use of Java objects or APIs to implement or marshall XML elements into XML schema that enforces domain and model data constraints is suggested in Kadel

packages (see TypeMetadataFactory, WMLtypeIO, WMLtypeRenderer, XMLTypeIO – Fig. 12A). Such similar use of Java APIs is also disclosed in Severin.

Severin discloses association of metadata describing the constraints for one to implement Java code based on such meta-information (para 0139-0148, pg. 11-12) including integration code for *marshalling* and *unmarshalling* XML (para 0103-0107, pg. 8), where Java interfaces implement validating and enforcing constraints and relationship (Constraint Implementation, Constraint Handler, Constraint inst Fig. 33;Constraint instance – Fig 6-7; precondition, postcondition Fig. 11; Fig. 17, 19), with using an "implementer" in Java code/interface object as instance for constraint validating. It would have been obvious for one skill in the art at the time the invention was made to implement the meta-information or validation function in Kadel so that the exposed Java objects are submitted (or used as inputs) into a implementation step where a API can be created (e.g. Kadel's instantiating a bean) using the Java objects and the constraint information as taught in Severin, so that Java interface objects (or APIs or bean instance) can include a functionality to marshall XML (in light of Severin) within a integration environment wherein XML meta schema represents domain's schema (set forth above by Kadel's API packages and reflection objects) and represent language used to validate or to enforce constraints as contemplated in both Kadel and Severin, because using the very (code construction) constraints or relationship information imposed by the metamodel as taught above(e.g. by Severin or Kadel use of Corba APIs) would enable the modeled application being target within an *integration* endeavor (as in Severin) to achieve a error-free internal infrastructure (e.g. indicating or notifying state of container or a context with respect to required or changed state of

its attribute or events - see Kadel: Fig. 33F, G, H, I,) when its constituting elements fulfill the construction requirement set forth by the above metamodel.

Kadel does not explicitly teach API including interface layer, proxy layer, and state layer. Kadel, however, discloses implementing a beans API but does not specify wherein the API comprises an interface layer, a proxy layer, and a state layer. Analogous to Kadel distributed network with use of J2EE and CORBA methodology (para 0311-0312 pg. 25-26) and Proxy interfaces (see Kadel: *ValidTestProxy* – Fig. 12A), Yeluripati teaches Web client/server session (Fig. 8) and creation of "functional bean" based on the EJB architecture including bean state (col. 4 li. 12-42), a proxy role(Fig. 6), and acting as an interface (col. 3 li. 47-62). It would have been obvious for one skill in the art at the time the invention was made to implement the XIS-based creation of API or beans in Kadel (*Java interfaces ... instantiate the appropriate bean* - para 0349-0350 pg. 30) so that such API would include a *interface* layer, a *proxy* layer and a *state* layer according to the above functionalities in EJB (as being applied by Yeluripati), because as implemented in this variety, this API would include all the needed service functionalities or layers as to be able to be called upon to handle the session and interchange between client, servers and the middle services like Corba, a NW paradigm using bean container methodology which Yeluripati also endeavors.

As per claim 3, Kadel discloses wherein the second language comprises XML (refer to the rationale in claim 1 addressing XMI/XML deriving of Java objects).

As per claim 4, Kadel discloses wherein the first language comprises UML (refer to claim 1).

As per claims 6-7, Kadel discloses wherein the first language comprises a customizable extension (e.g. Fig. 3A; addOneOfNService – Fig. 3B; Fig. 5; 36C-36D; para 0136 pg 9; Fig. 29); wherein the customizable extension is used to implement an additional feature of the API (refer to claim 1 based on addPluginService – Fig. 29).

As per claim 8, Kadel does not explicitly disclose wherein the additional feature comprises an indication of a file border. Kadel discloses API for JPanel package that operates on GUI component in terms of resizing, repainting, reshaping, paint Border, set Bounds, set Opaque (see JComponent, awtContainer, awt.Component - Fig. 33E) hence the identification of Gui file border in order to manipulate its graphic content is disclosed. And it would have been obvious for one skill in the art at the time the invention was made to implement the java libraries in view of the user manipulation, so that a feature included in the API would include a file border as set forth from the above, because this would help identify the target file upon which *awt* operation or painting methods would be defined.

As per claim 9, Kadel does not explicitly disclose wherein the API comprises a copy and paste operation. Kadel discloses XIS framework enabling editing of commands on GUI componetns, whereby the user can instantiate operation provided by the JAF API (see para 0349-0359, pg. 30; *canPaste()* Fig. 33b; *cut(clipboard)* Fig. 33c); and based on the copy-and-paste nature of the user operations to manipulate metadata attributes pertinent to a source/consumer scenario (see *cut and paste* - para 0335-0344, pg. 29) and to translate the user-customized parameters in a Java code procedure, it would have been obvious for one skill in the art at the time the invention was made to implement the XIS framework so that metadata and exposed Java classes libraries in view of JAF API (Fig. 33) are combined to support the creation of API

type of operation to actually edit the attributes or manipulate exposed meta hierarchy using the standard GUI fabric, because this would constitute efficient use of metadata and reusable Java packages whose utilization would be consistent with the extensibility aspect of the XIS framework, the extensive editing role played by user (Fig. 37A-D), and the availability of JAF API as set forth above.

As per claim 10, Kadel discloses a computer program product, tangibly embodied in an information carrier, for developing an application, the computer program product being operable to cause data processing apparatus to:

receive a first model in a first language, the first model defining development objects representing building blocks for developing the application, relationships among the development objects, and constraints for developing the application, wherein the first language comprises unified modeling language (refer to claim 1);and

generate set of intermediate objects as inputs to implement a API that enables implementing the development objects (refer to claim 1).

Kadel does not explicitly disclose convert the first model to a second model in a second language wherein the second model is an XMI model and the second language comprises XML; generate a set of intermediate objects by parsing the second model using an XML parser; but the limitations such as second model and deriving objects therefrom using a XML parser have been addressed as a combined rationale in claim 1.

Nor does Kadel explicitly disclose generating XML marshalling code, and an XML schema using the set of intermediate objects as inputs such that the XML schema enforces the

relationships and the constraints defined in the first model and enables implementing development objects. But this limitation has been addressed in claim 1.

Nor does Kadel explicitly disclose generating a proxy layer, a state layer, using the set of intermediate objects as inputs. But this limitation has been addressed in claim 1.

As per claim 12, refer to the rationale in claim 1 and 10 for the XMI/XML limitation.

As per claim 14, Kadel discloses wherein the set of intermediate objects comprises Java objects (refer to claim 10).

As per claim 15, Kadel discloses (by virtue of XMI, domain schema and XML derivation from XMI, as set forth in claim 10), wherein the XML schema includes a tree based on aggregation relationships in the first model (e.g. UML).

As per claims 16-17, Kadel does not explicitly disclose wherein the XML schema includes a reference based on an association relationship in the first model, and wherein the XML schema includes a complex type extension based on an inheritance relationship in the first model. UML as shown in Kadel includes association relationship and inheritance relationship (Fig. 3B; para 0080, pg. 5 para 0198-0200, pg. 15) when exposing meta-attributes related to the source/consumer model and data dependency flow. Based on Severin meta integration requiring mapping of complex association with need for new type creation (complex , new type - para 0086, 6; para 0186) among UML type hierarchies, it would have been obvious for one skill in the art at the time the invention was made to implement the XML schema intended to be reused in a XIS framework so that reference to association relationship to a UML and complex type extension are also represented in order to address the type extension and association needed within defined UML hierarchy, as by the mapping and integration (as in Severin) wherein

integreting the schema would need to address complex processes requiring extension into new complex types.

As per claim 18, Kadel discloses a computer program product, tangibly embodied in an information carrier, for developing an application, the computer program product being operable to cause data processing apparatus to:

receive a data model defining development objects representing building blocks for developing the application relationships among the development objects, and constraints for developing the application (refer to claim 1),

derive an API based on the set of intermediate objects (refer to claim 1)

and use the API to perform operations on the development objects (para 0349-0350 pg. 30; para 0129-0135 pg. 9).

Kadel does not explicitly disclose: *generate an XMI model that is a representation of the data model; generate a set of intermediate objects by parsing the XMI model using an XML parser*. But the limitations such as second model and deriving objects therefrom using a XML parser have been addressed as a combined rationale in claim 1.

Nor does Kadel explicitly disclose wherein the API comprises XML marshalling code, and an XML schema to enforce the constraints but this API included XML schema and marshalling code have been addressed in claim 1.

Nor does Kadel explicitly disclose wherein the API comprises an interface layer, a proxy layer, a state layer; but this limitation has been addressed in claim 1.

As per claim 23, Kadel (by virtue of Severin) discloses wherein the first model is stored one storage module (schematized structures -- representing a UML model, imported into a XIS

framework – see Kadel: para 0083 pg. 5- reads on UML first model document being stored in a file system of a framework or integration memory).

As per claim 24, refer to claim 23.

As per claims 25-26, Kadel discloses wherein the set of intermediate objects comprises Java objects (refer to claim 14).

6. Claims 20-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kadel et al, USPubN: 2002/0184401; Severin, USPubN: 2005/0005251; Yeluripati et al, USPN: 7,086,065; further in view of Hejlsberg et al, USPN: 6,920,461 (hereinafter Hejlsberg)

As per claims 20-21, Kadel does not explicitly disclose operations to include:

creating a new development object as a transient object without an existing corresponding file(e.g. para 0312 pg. 26 – Note: creating of template then use it for adding/deleting Fig. 29-31; Fig. 33 – reads on transient object without a persistent file)

Kadel does not explicitly disclose modifying the transient object until the transient object is committed to a persistent file; and to destroy the transient object if a delete command is requested before the transient object is committed to a persistent file.

Kadel discloses code implemented to match SQL queries using XML elements to instantiate application to interface with database (para 0300-0310, pg. 25) where code to implement requires pluggable service and attribute conversion using JAF collaboration of classes with editing capabilities (Fig. 29-31; Fig. 33) wherein user's deleting, adding of data is effectuated via a template usage (para 0312 pg. 26) all of which data being temporary until determination to commit such implementation. Using a development application interface (analogous to Kadel) operating on layers or namespaces that expose class libraries or

enumeration of related data structures or code constructs or tables (see Hejlsberg: col. 6; Fig. 2) Hejlsberg discloses application code instantiation from the libraries of reuse classes or OO packages (e.g. C++, Jscript, Microsoft ".NET" APIs) including UI objects with procedures to save a view, to customize drawing or drag-drop (col. 7, lines 48-62; col 8 lines 22-50) and a SQL namespace to interface with a database (col. 8 line 50 to col 9 line 11) including procedures to validate proper constructs, for implementing operations as to commit, dispose, rollback, save, accept/reject changes, cancel Edit (RejectChanges, acceptChanges – col 55; commit, delete, rollback – col. 57; CancelEdit col. 64; Delete, AcceptChanges – col. 65; Dispose, Finalize – col. 289; Commit, Dispose, Rollback, Save - col. 326). Based on methods based on reuse libraries to implement API in terms of constraints as in Severin and Kadel and the intended framework enabling users to decide whether how to add/remove or commit template/transient data/objects, it would have been obvious for one skill in the art at the time the invention was made to implement the code or APIs based on core libraries as practiced in both Kadel and Hejlsberg, such that a transient object in the process of validating data, information, and implementation details as taught in Kadel's user-driven customization approach (e.g. using template) would be supported by capability to create APIs with methods to destroy a transient object or to commit it to a persistent form, as taught in Hejlsberg from above. One would be motivated to do this (i.e. create APIs by the user to destroy a transient object if it is not made for persistence committing) because that way the created APIs would enable changes caused by a customization view in Kadel's approach to detect errors prior to commit, and allowing removal of undesired implementation gathering of data, whereby obviate potential runtime errors should actual translation of uncorrected constructs become finalized.

As per claim 22, Kadel does not explicitly disclose instructions to mark the persistent file as deleted if a delete command is requested after the transient object is committed to a persistent file. Based on Hejlsberg's DB-related method to indicate that change data is not accepted or that a transient form of changes is committed, or to rollback otherwise (e.g. RejectChanges, acceptChanges – col 55; commit, delete, rollback – col. 57), the notion of keeping a change with persisting of a accepted version along with removing an older version is suggested. Hence, this method as to mark an older file/record as deleted after a persisting operation has been completed (by creating a new file) would have been obvious in view of the requirement to reconcile persisted data (e.g. DB records, not keeping two records with same identification) rationale as set forth above.

Response to Arguments

7. Applicant's arguments filed 8/31/09 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

USC § 102 Rejection

(A) The arguments (Appl. Rmrks pg. 10) are considered largely moot in view of the current grounds of rejection being necessitated by the Amendments.

USC § 103 Rejection

(B) Applicants' argument is deemed founded on claim language (Appl. Rmrks pg. 12) that belongs to the current Amendment, which was not recited in the previously prosecuted claims. Therefore, the arguments cannot establish how these newly added limitations would have been non-obvious in light of the previous Office Action, which is herein supplanted by new and necessitated grounds of Rejection.

The claims stand rejected as set forth in the Office Action.

(C) Following are some remarks of the Examiner directed towards a claim language possibility in hope for establishing a start of 'allowable subject matter'.

The XMI (second model) and parsed XML (first XML) should be taught in the claimed scenario to meaningfully support the derivation of Java (intermediate) objects, those OO objects which in turn would be instrumental in creating a more global "metadata API", this "metadata API" which at runtime would be operable as to be built from, or include or fetch specific object oriented APIs to **not only** validate meta-information or its syntax, **but also** generate further XML form (second XML distinct from the first XML), which require "API" instantiated code to marshall (e.g. as this second XML data is loaded onto target integration environment) wherein the very first XMI derived XML and intermediate Java (intermediate) code (intermediate API constructs) in combination supports the framework use of "metadata API" (non-intermediate APIs) to validate constraints of target model (e.g. UML or reusable XML-based model) or unmarshall/marshall further (second XML) metadata needed for multi-tiered or NW target runtime. Further, the preamble to all claims has to put forth a 'metadata API' framework including use of initial first model to obtain intermediate objects (via XMI) that generate and retrieve the second XML for a intended development; with **emphasis on** a) clear sequence relationship between the actions being claimed, and b) proper mapping of claim limitations with portions of the Specifications.

Conclusion

8. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished

applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

December 11, 2009